

José Valentim dos Santos Filho^{*,†,ⓐ 1}

Directeurs de thèse: Aloysio PEDROZA* et Jean-Pierre COURTIAT[†]

Laboratoire d'accueil:

* GTA - Grupo de Teleinformática e Automação
Centro de Tecnologia, Sala H301, Ilha do Fundão, CEP: 21945-970, Rio de Janeiro - Brésil

Etablissement d'inscription:

Univ. Federal do Rio de Janeiro

[†] LAAS - Laboratoire d'Analyse et d'Architecture des Systèmes UPS - Université Paul Sabatier
7, Av. du Colonel Roche 31077 Toulouse Cedex 4 118, Route de Narbonne 31062 Toulouse

[ⓐ] Étudiant financé par la Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - *CAPES*,
Brésil.

Résumé

Un des problèmes critiques concernant le déploiement d'applications coopératives consiste à offrir à chaque utilisateur une vraie «notion de présence» des autres utilisateurs avec lesquels il coopère. La réalité virtuelle, en permettant de retranscrire le monde réel par des métaphores, rend cette «notion de présence» naturelle, avec des possibilités d'interaction très proches du réel. Nous proposons ici un CVE (Collaborative Virtual Environment) permettant à un ensemble de personnes de travailler en groupe dans un monde virtuel partagé. Dans ce CVE, des mécanismes génériques sont définis dans le but de permettre à des avatars (représentant les utilisateurs) d'interagir avec des objets et d'autres avatars placés dans le même monde virtuel, permettant ainsi aux utilisateurs de visualiser le plus explicitement possible les activités de coopération en cours.

Mots clés

CVE, Interaction, Connaissance, STEP, Actions, Dynamique

1 INTRODUCTION

Dans cet article nous allons présenter un CVE qui est un enrichissement du CVE VNet². Dans ce contexte, nous nous sommes plus intéressés à la définition des mécanismes génériques qui permettent de réaliser des interactions entre des avatars et des objets placés dans un monde virtuel et à l'intégration du CVE à des applications coopératives qu'aux problèmes spécifiques des CVEs (propagation d'états, coordination, cohérence, synchronisation, etc.)

Par ailleurs, nous avons défini une architecture de CVE pour donner aux avatars la connaissance exacte des objets qui sont dans un monde virtuel, ainsi que de toutes les actions qu'ils peuvent exécuter sur chacun de ces objets. Cette architecture offre aussi la possibilité de composer dynamiquement des actions pour les avatars, à chaque fois qu'un nouvel objet est ajouté au monde virtuel. La capacité des avatars d'exécuter une action spécifique sur un objet nous permet d'exprimer avec plus de clarté les besoins de coopération d'un utilisateur, et également d'extraire de chaque action une sémantique qui

1. valentim@laas.fr

2. <http://www.csclub.uwaterloo.ca/u/sfwhite/vnet>

peut être exploitée au delà du contexte d'un seul monde virtuel. Par ailleurs, la définition d'une API permettant de recevoir et d'envoyer des événements du monde virtuel à des applications coopératives externes est très fortement souhaitable et a été prévue dans notre architecture.

Cet article est structuré de la manière suivante. La section 2 présente un état de l'art des environnements virtuels collaboratifs. Dans la section 3, nous décrivons en détail l'architecture du CVE ici proposé. Finalement, dans la dernière section, nous donnons quelques conclusions et directions de travail futur.

2 ÉTAT DE L'ART

Les CVEs constituent depuis quelques années un sujet de recherche en plein essor, et plusieurs technologies pour le développement d'environnements virtuels collaboratifs ont été proposées et décrites dans la littérature. Plusieurs efforts ont en particulier été réalisés pour traiter des problèmes inhérents à la nature répartie des CVEs.

COSMOS-2 [1] vise à définir un cadre conceptuel général pour la mise en oeuvre d'applications collaboratives virtuelles. *COSMOS-2* est développé en Java et utilise la bibliothèque *Java's Java3D* (pour le rendu des objets 3D) et le *Java Media Framework* (JMF) pour réaliser l'intégration de capacités vidéo et audio à l'application. Le monde virtuel peut être modifié par les utilisateurs (par exemple, chargement de nouveaux objets dans une scène virtuelle à partir d'un répertoire local) et la synchronisation du monde virtuel est réalisée par l'intermédiaire d'une session *multicast*.

MASSIVE-3 [2] est un environnement virtuel de collaboration qui supporte des communications audio temps-réel entre utilisateurs et la modification dynamique des mondes virtuels. Le système est disponible sous forme d'une implémentation s'appuyant sur une librairie (classes C++) et les utilisateurs peuvent utiliser cette bibliothèque pour développer leur propre CVE.

NPSNET-V [3], au lieu de disposer d'une bibliothèque applicative, propose un cadre général (*framework*) pour développer des mondes virtuels en réseau, complètement répartis, persistants et à base de composants. *NPSNET-V* permet d'ajouter des composants applicatifs lors de l'exécution, grâce au chargement dynamique de classes Java.

Dans le cas de *Dive* [4], plusieurs acteurs peuvent entrer et quitter les mondes virtuels de manière dynamique. Un acteur peut être une personne ou un processus applicatif automatisé. Les processus applicatifs peuvent créer et introduire les objets graphiques nécessaires; ils sont également capables d'écouter et de réagir aux événements produits à l'intérieur du monde virtuel.

Ces différentes solutions traitent des problèmes importants spécifiques aux CVEs. Cependant, rien dans l'architecture de ces CVEs, ne prévoit de manière satisfaisante de réaliser des interactions entre avatars et objets au sein d'un même monde virtuel, ni l'intégration du CVE avec d'autres applications coopératives externes.

Dans le cas de systèmes qui s'appuient sur des agents, comme le système Agile [5], les agents autonomes ont des émotions, des personnalités et des intelligences. Ils perçoivent l'environnement et réagissent selon des règles prédéfinies. Le projet *InViWo* [6] est un autre exemple de système qui s'appuie sur des agents autonomes. Comme avec *Agile*, les agents *InViWo* perçoivent l'environnement et réagissent selon des règles de comportement définies en utilisant le langage proposé, *Marvin*.

Dans *PAR* [7], l'utilisateur peut donner des ordres aux avatar au moyen de commandes textuelles, par exemple, "*Sarah, walk to the door*" et alors un module analyse et évalue l'applicabilité de cet ordre

vis-à-vis des objets et des avatars présents dans la scène virtuelle.

Notre travail se distingue de ceux décrits antérieurement car dans l'architecture de CVE que nous proposons, un avatar peut accumuler des connaissances sur des nouveaux objets ajoutés au monde virtuel en temps d'exécution. Ce qui signifie qu'au fur et à mesure que les objets sont ajoutés au monde virtuel, l'utilisateur peut définir de nouvelles actions appropriées à ces objets et il peut également, composer d'autres actions à partir des actions déjà existantes.

3 NOTRE PROPOSITION DE CVE

Notre modèle d'avatar s'appuie sur le standard *H-Anim*³ qui est un standard pour la description d'humanoïdes animés. Un corps *H-Anim* est une hiérarchie de noeuds d'articulation auxquels peuvent être associés d'autres noeuds d'articulation.

Ainsi, faire tourner une partie du corps d'un humanoïde consiste à attribuer de nouvelles valeurs de localisation aux noeuds d'articulation concernés. Cela a pour conséquence que définir une action en utilisant le standard *H-Anim* est une tâche assez compliquée et non intuitive, à cause de la distance existante entre *H-Anim* (langage d'animation) et le langage naturel. Pour résoudre ce problème, plusieurs langages de script, ayant pour vocation de devenir des standards, ont été proposés, tels que *VHML* [8], *CML* et *AML* [9] et *MPML* [10] qui s'appuient tous sur la norme *XML*. Ces différents langages ne s'appuient par contre pas sur les standards *VRML/X3D* [11][12], ni sur *H-Anim*.

STEP [13], est un langage de script qui s'appuie sur les standards *VRML/X3D* et *H-Anim*; il fournit une interface disposant d'un riche ensemble d'opérateurs qui facilite la définition des actions, grâce à la proximité de *STEP* au langage naturel. Par exemple, en *STEP*, l'action "turning the left-arm forward slowly" peut être spécifiée par l'opérateur suivant:

```
turn(Agent, l_shoulder, front, slow)
```

Le noyau de *STEP* traduit alors le script décrivant l'action en commandes (*H-Anim*) qui seront exécutées par *EAI VRML/X3D* [14], comme cela est illustré dans la figure 1. L'*EAI* est une norme du standard *VRML* qui définit une interface entre le monde virtuel (*VRML*) et un environnement externe.

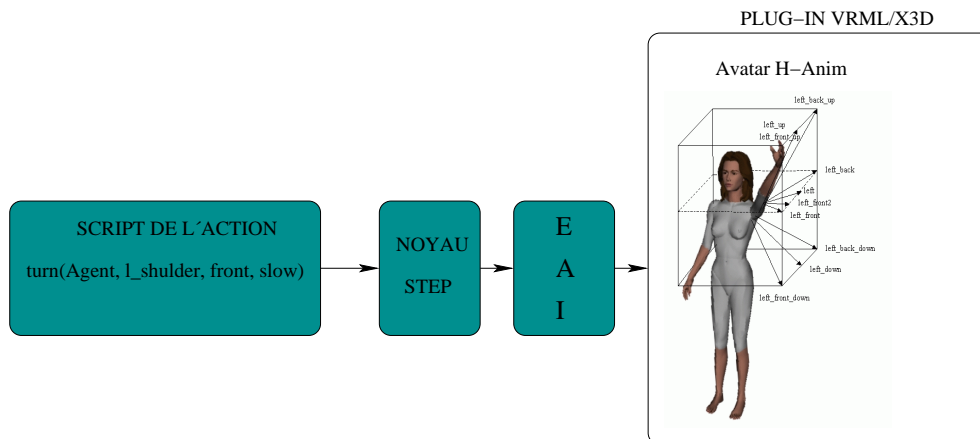


FIG. 1 – Animation d'un avatar *H-Anim* en utilisant le langage de script *STEP*

3. <http://www.h-anim.org>

Les operateurs suivants sont également fournis par *STEP*:

- `move(Avatar, BodyPart, Direction, Duration)` - pour déplacer la partie *BodyPart* du corps de l'avatar;
- `repeat(Action, T)` - pour répéter *T* fois l'action appelée *Action*;
- `seq([Action1,...,Actionn])` - pour composer séquentiellement les actions *Action₁*,..., et *Action_n*;
- `par([Action1,...,Actionn])` - pour composer en parallèle les actions *Action₁*,..., et *Action_n*.

En utilisant ces différents opérateurs d'action, l'utilisateur peut définir des actions individuelles qui sont définies comme des actions associées à l'avatar lui-même, telles que «marcher», «courir», «sauter», etc. Ces opérateurs permettent également de composer une nouvelle action à partir d'autres actions déjà existentes. Par exemple, si les actions «marcher» et «s'asseoir» existent déjà, l'utilisateur peut utiliser l'opérateur *par* pour composer une nouvelle action «pédaler»⁴:

```
script(pédaler(Avatar), Action):-  
Action = par([marcher,s'asseoir]).
```

Ou encore, en utilisant l'opérateur *par* et l'opérateur *move*, composer l'action «monter_escalier» à partir de l'action «marcher»:

```
script(monter_escalier(Avatar,Marches), Action):-  
Action = par([marcher,move(Avatar,up)]).
```

Dans ce cas, les marches de l'escalier constituent un paramètre qui doit être spécifié par l'utilisateur au moment d'exécuter l'action.

La capacité de composer dynamiquement les actions des avatars offre aux utilisateurs la possibilité de soumettre ces avatars à un processus d'apprentissage du monde virtuel en temps d'exécution. C'est-à-dire, à chaque fois qu'un nouvel objet est ajouté au monde virtuel, l'utilisateur peut définir ou composer des nouvelles actions appropriées à cet objet. Ces scripts définissant les actions sont enregistrés dans un répertoire local pour être utilisés ultérieurement.

Le fait que les scripts soient instanciés avec comme paramètre le nom de l'avatar, signifie qu'un script peut être exécuté sur n'importe quel avatar construit selon le standard *H-Anim*. Donc, un utilisateur peut composer une action appropriée à un objet et l'envoyer (le script définissant l'action) à d'autres utilisateurs raccordés au monde virtuel, qui pourront ainsi faire ses respectives avatars exécuter cette action.

3.1 ARCHITECTURE

À l'heure actuelle, l'architecture du CVE proposée reste centralisée (comme dans la proposition originale de Vnet). Sur la figure 2, nous pouvons voir ses différentes composantes et la manière dont elles sont agencées.

Nous allons maintenant étudier plus en détail chaque composante de l'architecture proposée.

4. Nous n'allons pas entrer dans le détail de la définition des actions

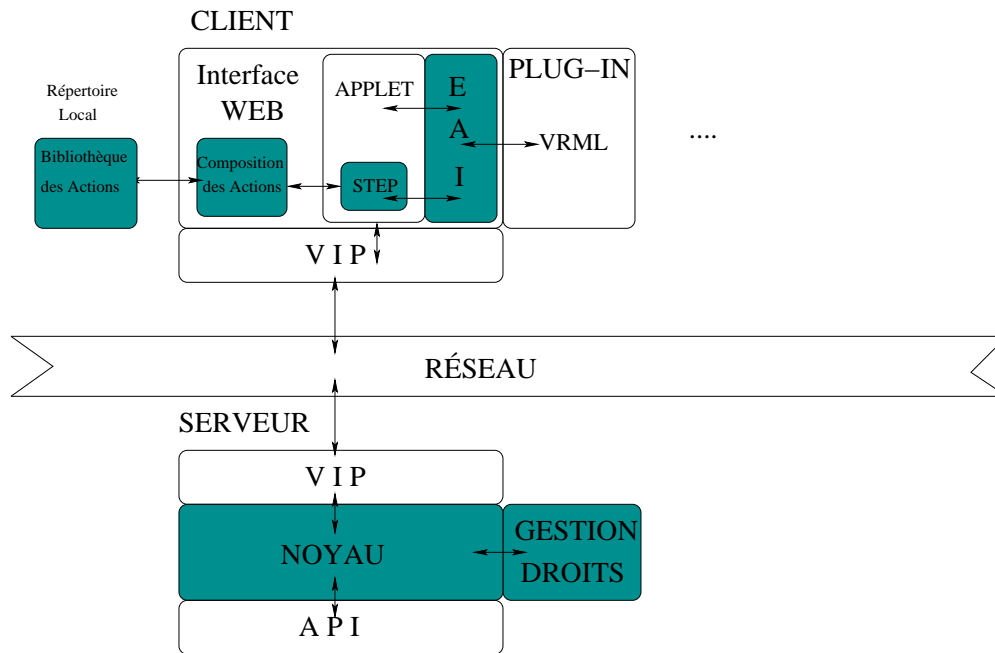


FIG. 2 – l'Architecture du CVE

3.1.1 Le client

Chaque client est constitué d'une interface WEB, d'un navigateur VRML qui affiche la scène virtuelle dans laquelle évolue l'utilisateur et d'un *applet* JAVA qui contient un module STEP. Le CVE utilise, au moins pour l'instant, le protocole *VIP - VRML Interchange Protocol*, pour assurer la communication entre un client et le serveur. Dans un premier prototype, *VIP* sera étendue avec quelque *PDU (Protocol Data Unit)* appropriées à la définition des actions des avatars

L'interface WEB permet à l'utilisateur de se raccorder au monde virtuel en étant représenté par un avatar *H-Anim*, de naviguer dans ce monde en utilisant des actions individuelles, déjà fournies par le module *STEP*, et d'ajouter des objets dans le monde virtuel. Au moment d'ajouter un nouvel objet dans le monde virtuel, l'utilisateur doit associer un ensemble d'actions à l'objet. Il peut, choisir une ou plusieurs actions déjà existantes dans son répertoire local et qui sont présentées au niveau de son interface, ou composer une nouvelle action à partir d'actions déjà existantes, ou finalement définir une toute nouvelle action. Puis, un message *VIP* est construit qui inclut l'identificateur de l'objet, sa localisation et l'ensemble des actions associées à cet objet; le message *VIP* est envoyé au module de gestion des droits, comme cela est illustré dans la figure 3. Le module de gestion des droits est présent dans le serveur, comme cela sera décrit dans le paragraphe 3.1.2. L'information (objet, actions disponibles sur cet objet) est enregistrée dans le module de gestion des droits pour être consultée dès que cet objet est sélectionné par un utilisateur.

Un simple système de communication textuelle est également fourni par l'interface.

3.1.2 Le serveur

Lorsqu'un client se raccorde au système, le serveur doit mettre à jour la base de données persistante contenant les informations concernant les clients raccordés. Il doit ensuite informer les autres clients de ce changement afin de modifier l'état du monde virtuel chez ces derniers.

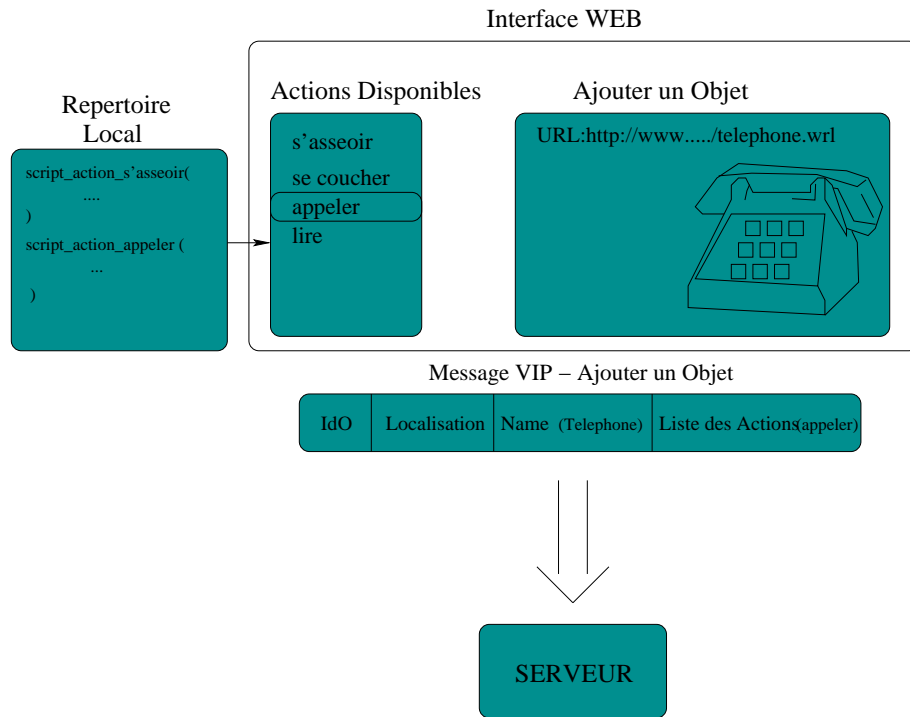


FIG. 3 – Ajout d'un Objet dans le monde virtuel

Gestion de droits

VNet, originalement, n'a aucun contrôle sur les droits des utilisateurs, ce qui signifie que n'importe quel utilisateur peut supprimer du monde virtuel l'avatar d'un autre utilisateur. Deblasi [15] a créé, pour VNet, un module de gestion des droits des utilisateurs, séparé en trois catégories correspondant à trois aspects de la gestion du monde virtuel: les droits relatifs aux entités qui couvrent les possibilités d'interaction des utilisateurs vis-à-vis des objets; les droits des utilisateurs vis-à-vis du monde virtuel global; et finalement les rôles que peuvent prendre les utilisateurs.

Nous introduisons dans le module de gestion des droits, le concept de droit des avatars. Ce concept est créé juste pour différencier le contrôle sur les actions exécutées par les avatars de celles exécutées par les utilisateurs.

À chaque fois qu'un utilisateur sélectionne un objet, un message VIP contenant l'identification et le rôle de l'utilisateur, l'identification de son avatar et l'objet sélectionné, sera construit et envoyé au module de gestion des droits. Le gestionnaire des droits à son tour, évalue quels sont les droits d'accès que l'utilisateur dispose sur l'objet sélectionné. Dans le cas d'une évaluation positive, un message VIP contenant l'identification de l'avatar, la localisation de l'objet et l'ensemble des actions disponibles pour cet objet sera construit et envoyé au client pour que l'utilisateur puisse choisir quelle action réaliser.

Api

La définition d'une API pour le CVE donne la possibilité de collaborer au delà du monde virtuel. Les actions des utilisateurs et des avatars sont traduites, respectivement, par des événements réels et virtuels qui seront passés à des systèmes d'intégration d'outils collaboratifs, comme par exemple, le système *LEICA* [16] qui est en cours de développement au sein du groupe OLC du LAAS/CNRS⁵. Pour

5. <http://www.laas.fr>

la définition des fonctionnalités que le CVE mettra à disposition du système d'intégration, nous allons prendre en compte les actions les plus significatives des utilisateurs et des avatars dans le contexte de collaboration. Par exemple, sélectionner l'avatar d'un utilisateur en démarrant automatiquement une conférence audio; ou bien un objet virtuel représentant un téléphone pourrait se comporter comme un lien vers une application VoIP, permettant à des utilisateurs raccordés au monde virtuel d'appeler des personnes qui sont en dehors de leur monde virtuel, tout en utilisant des métaphores du monde réel. Cet esprit nous guidera dans le choix des fonctionnalités disponibles.

4 CONCLUSION ET PERSPECTIVES

Dans cet article, nous avons proposé l'architecture d'un CVE qui donne aux avatars la connaissance de ce qu'ils peuvent faire avec les objets dans un monde virtuel. Les avatars ont aussi la capacité d'apprendre dynamiquement l'existence de nouveaux objets ainsi que les actions définies pour ces objets. L'architecture du CVE est, à l'heure actuelle, centralisée, mais l'évolution vers une architecture répartie est envisagée. Vu que le protocole utilisé pour assurer la communication entre les clients et le serveur, VIP, n'est pas compatible avec *X3D*, la définition d'un nouveau protocole est nécessaire. Nous allons ainsi évoluer vers un protocole compatible avec *X3D* et qui puisse être étendu pour décrire les actions effectuées par les avatars sur les objets. Le protocole *XFSP (Cross-Format Schema Protocol)* [17] est une possibilité à considérer.

Références

- [1] V. Darlagiannis et N. Georganas, "Virtual collaboration and media sharing using COSMOS", in *Actes de 4th WORLD MULTICONFERENCE CSCC'00, Greece, Juillet 2000*.
- [2] C. Greenhalgh, J. Purbrick et D. Snowdon, "Inside massive-3: flexible support for data consistency and world structuring", in *CVE '00: Proceedings of the third international conference on Collaborative virtual environments*, pp. 119–127, ACM Press, 2000.
- [3] M. Capps, D. McGregor, D. Brutzman et M. Zyda, "Npsnet-v: A new beginning for dynamically extensible virtual environments", in *IEEE Computer Graphics and Applications*, vol. 20, pp. 12–15, 2000.
- [4] E. Frécon et M. Stenius, "DIVE: A scalable network architecture for distributed virtual environments", in *Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments)*, vol. 5, pp. 91–100, 1998.
- [5] Y. Zhang, L. Guo et N. D. Georganas, "AGILE: An architecture for agent-based collaborative and interactive virtual environments", in *Proc. Workshop on Application Virtual Reality Technologies for Future Telecommunication System, IEEE Globecom'2000 Conference*, 2000.
- [6] N. Richard, "InViWo agents: Write once, display everywhere", in *Proceedings of the Web3D Conference*, 2003.

- [7] N. I. Badler, R. Bindiganavale, J. Allbeck, W. Schuler, L. Zhao et M. Palmer, "Parameterized action representation for virtual human agents", *Embodied conversational agents*, pp. 256–284, 2000.
- [8] A. Marriott et J. Stallo, "VHML - Uncertainties and Problems. A discussion...", *AAMAS-02 Workshop on Embodied Conversational Agents - Let's Specify and Evaluate Them!*, 2002.
- [9] Y. Arafa, K. Kamyab, E. Mamdani, S. Kshirsagar, N. Magnenat-Thalmann, A. Guye-Vuillème et D. Thalmann, "Two Approaches to Scripting Character Animation", *AAMAS-02 Workshop on Embodied Conversational Agents - Let's Specify and Evaluate Them!*, 2002.
- [10] H. Prendiger, S. Descamps et M. Ishizuka, "MPML: a markup language for controlling the behavior of life-like characters", *Journal of Visual Languages and Computing*, 2004.
- [11] "VRML (ISO/IEC 14772-1), virtual reality modeling language", 1997.
- [12] "X3D (ISO/IEC FDIS 19775:200x), X3D framework & SAI - Final Draft International Standard", 2004.
- [13] Z. Huang, A. Eliëns et C. Visser, "STEP: A Script Language for Embodied Agents", in *PRICAI-02 International Workshop on Lifelike Animated Agents. Tools, Agents and Applications* (H. Prendiger, ed.), pp. 46–51, 2002.
- [14] "VRML EAI (ISO/IEC FDIS 14772-2), the vrmf external authoring interface committee final draft", 2002.
- [15] A. Deblasi, "Interfaces virtuelles pour le travail collaboratif", in *Diplôme D'Études Approfondies - ENSICA*, 2003.
- [16] R. L. Gomes, G. de Jesús Hoyos Rivera et J.-P. Courtiat, "LEICA: Loosely-Coupled Integration of CSCW Systems", in *5th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2005) LNCS Springer-Verlag, Athens, Greece*, Juin 2005.
- [17] D. Brutzman et D. McGregor, "XML binary serialization using cross-format schema protocol (XFSP) and XML compression considerations for extensible 3d (X3D) graphics", in *W3C Workshop, USA*, Septembre 2003.